Available online at www.sciencedirect.com
SciVerse ScienceDirect
Procedia Computer Science 6 (2011) 69–75
Procedia Computer Science

Complex Adaptive Systems, Volume 1
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2011- Chicago, IL

A Combination of Shuffled Frog Leaping and Fuzzy Logic for Flexible Job-Shop Scheduling Problems

Wannaporn Teekeng*, Arit Thammano

Computational Intelligence Laboratory, Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520 Thailand

Abstract

Flexible Job-Shop Scheduling Problem (FJSP) is a well known NP-hard combinatorial optimization problem. Over the last decade, many algorithms have been proposed to tackle FJSP. Evolutionary algorithms, which solve problems by mimicking the process of natural evolution, are the most widely used techniques in solving FJSP. This paper proposes a novel evolutionary algorithm that integrates the concept of a fuzzy logic into Shuffled Frog Leaping Algorithm. In the proposed SFLA-FS model, the fuzzy roulette wheel selection is used in selecting frogs to form a sub-memeplex. This selection method is proved to be better than the typically used rank selection. The objective of this research is to find a schedule for each of the 10 benchmark problems that minimize their makespan. The experimental results obtained from SFLA-FS show that the SFLA-FS is very efficient for all tested problems.

Keywords: Flexible Job-Shop Scheduling; Fuzzy Logic; Shuffled Frog Leaping Algorithm; Fuzzy Roulette Wheel

1. Introduction

Scheduling problems occur in many real-world applications, e.g. manufacturing, supply chain planning, and production line planning. Many artificial intelligence techniques have been proposed to find the optimal solutions of the problems. It is well known that the Job-Shop Scheduling Problem (JSSP) is one of the hardest combinatorial optimization problems. The Flexible Job-Shop Scheduling Problem (FJSP) is an extension of the Job-Shop Scheduling Problem (JSSP). FJSP allows each operation to be processed on more than one machine. For smaller

* Corresponding author. Tel.: +66(0)2723-4900; fax: +66(0)2723-4910.
E-mail address: w_teekeng@hotmail.com.

1877–0509 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.procs.2011.08.015

problems, mathematical programming, such as Integer linear programming and Branch-and-bound algorithms, can be used to find the optimal solutions; however, the above methods cannot guarantee optimal results for larger problems. Recently, a number of metaheuristic approaches, such as Genetic Algorithm (GA) [1], Tabu Search [2], Simulated Annealing (SA) [3], Particle Swarm Optimization (PSO) [4], and other approaches including Fuzzy Logic [5] have gained a lot of attention from researchers in the area.

Shuffled Frog-leaping algorithm (SFLA) is a new addition to the range of intelligent algorithms and a new member to the family of memetic algorithms. The SFLA uses memetic evolution in the form of infection of ideas from one individual to another in a local search. The local search is similar in concept to a PSO algorithm and can search for food based on a colony. In addition, SFLA can search for global exploration and is easy to understand. In recent years, several modifications have been proposed: Solving TSP with Shuffled Frog-leaping Algorithm [6]; Improved Shuffled Frog-leaping Algorithm for continuous optimization problem [7]; A Novel Memetic Algorithm for Global Optimization Based on PSO and SFLA [8].

In this paper, we propose a novel evolutionary algorithm that integrates the concept of fuzzy logic into Shuffled Frog-leaping Algorithm. The novel fuzzy roulette wheel selection is proposed and used in selecting frogs to form a sub-memeplex. Additionally, the local search procedure is used to further improve the resulted solution.

This paper is organized as follows: Section 2 briefly describes the flexible job-shop scheduling problems and the original shuffled frog leaping algorithm. In Section 3, the proposed SFLA-FS algorithm is presented. The experimental results are given in Section 4. Section 5 provides the conclusion to the paper.

2. Related Theories

2.1. Flexible Job-Shop Scheduling Problems

The Flexible Job-Shop Scheduling Problem is an extension of the classic job shop scheduling problem which allows an operation to be processed by any machine out of a set of available machines. The objective of FJSP is to schedule a set of R jobs $J = \{J_1, J_2, \ldots, J_R\}$ on a set of S machines $M = \{M_1, M_2, \ldots, M_S\}$ so that the makespan (C_{max}) is minimized. Each job may have different number of operations. Each operation $O_{i,j}$, the j^{th} operation of the i^{th} job, can be processed on any of the allowable machines. An example of FJSP is given in Table 1. Each row refers to an operation; each column refers to a machine and cells are processing times. As illustrated in Table 1, for example, the 3^{rd} operation of the 2^{nd} job is only allowed to be processed on M_1 and M_3.

Table 1. Example of the Flexible Job-Shop Scheduling problem.

		Machines		
Job	Operation	M_1	M_2	M_3
	$O_{1,1}$	2	4	8
J_1	$O_{1,2}$	5	4	7
	$O_{1,3}$	2	-	-
	$O_{2,1}$	10	11	10
J_2	$O_{2,2}$	6	5	-
	$O_{2,3}$	2	-	6

2.2. The Shuffled Frog Leaping Algorithm

Shuffled Frog Leaping Algorithm (SFLA) is proposed by Eusuff and Lansey [9, 10]. The algorithm is inspired by the behavior of frogs when seeking for the food. The algorithm begins by randomly selecting F frogs and sorting them according to their fitness value order. Next, the frogs are divided into m memeplexes. For each memeplex, q frogs are randomly selected to form a sub-memeplex. The chance of being selected is proportional to the frog fitness; fitter frogs have a higher chance of being included in the sub-memeplex.

$$P_j = 2(n+1-j)/[n(n+1)] \quad (1)$$

where n is the number of frogs in each memeplex.

In each sub-memeplex, the worst frog is selected. Then, the position of the worst frog in each sub-memeplex is updated according to Eq. (2), where the worst frog learns to move from the best frog in its sub-memeplex.

$$Fworst^{k+1} = Fworst^{k} + r(Fsbest^{k} - Fworst^{k}) \quad (2)$$

where Fworst is the position of the worst frog in the sub-memeplex.
Fsbest is the position of the best frog in the sub-memeplex.
r is a random number between 0 and 1.
k is the iteration number of the sub-memeplex.

If the new position is better than the previous position, the previous position is replaced by the new position. Otherwise Eq. (3) is used to update the position of the worst frog instead.

$$Fworst^{k+1} = Fworst^{k} + r(Fgbest - Fworst^{k}) \quad (3)$$

where Fgbest is the position of the best frog in the population.

3. The FSLA-FS algorithm

This paper proposes a novel evolutionary algorithm that integrates the concept of a fuzzy logic into Shuffled Frog Leaping Algorithm (SFLA-FS). The detail of the algorithm is given below:

3.1. Generate the Initial Population

In this step, an initial population of F frogs $X = (x_1, x_2, \ldots, x_f, \ldots, x_F)$ is created. Each frog, which in this paper is defined as shown in Fig 1, represents a possible solution to a problem. The length of each frog is equal to the total number of operations. As illustrated in Fig 1, each meme consists of two components: the operation to be processed and its corresponding machine. The first components of the memes are initialized by randomly permute the operations to be scheduled. Then, starting from the first meme, the corresponding machine is selected for the operation by using the shortest processing time (SPT) scheme as shown in Fig 2.

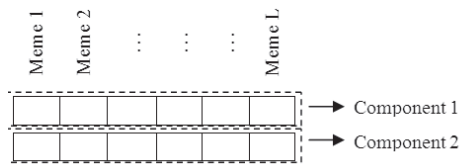

Fig. 1 The structure of a frog.

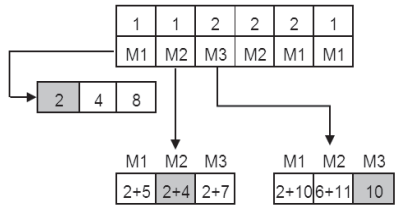

Fig. 2. Example of the process of assigning the machine to the operation.

3.2. Construct a sub-memeplex using the fuzzy roulette wheel selection

The F frogs are ranked in descending order. Then they are partitioned into m memeplexes. Each memeplex contains n frogs. For each memeplex, q frogs are selected to from a sub-memeplex. This paper proposed a new approach in selecting frogs to form a sub-memeplex. The main concept of the proposed approach is that the pair of frogs which are similar in their content are placed next to each other with some degree of overlap in the roulette wheel. For example, in Table 2, the population consists of 4 frogs; each frog represents a different possible solution. The following paragraphs show how to create the fuzzy roulette wheel.

Table 2. Some solutions of the example in Table 1.

Frog	Solutions
1	$O_{2,1}$, $O_{1,1}$, $O_{2,2}$, $O_{1,2}$, $O_{1,3}$, $O_{2,3}$
2	$O_{2,1}$, $O_{1,1}$, $O_{2,2}$, $O_{1,2}$, $O_{2,3}$, $O_{1,3}$
3	$O_{1,1}$, $O_{2,1}$, $O_{1,2}$, $O_{2,2}$, $O_{2,3}$, $O_{1,3}$
4	$O_{2,1}$, $O_{1,1}$, $O_{1,2}$, $O_{2,2}$, $O_{2,3}$, $O_{1,3}$

In this paper, the first frog is always assigned to the first slot of the roulette wheel. The slot size is proportional to its fitness value. Then the similarity between the first frog and the rest of the frogs (frogs 2, 3, and 4) are calculated. A pair of frogs with the highest similarity score is chosen; in this case, the second frog is selected with a similarity score of 4 (Fig 3(a)). Therefore, the second frog owns the second slot. The same process is repeated until all frogs are allocated slots in the roulette wheel. As illustrated in Figs 3(b) and 3(c), the fourth frog is assigned to the third slot while the third frog is assigned to the fourth slot of the roulette wheel.

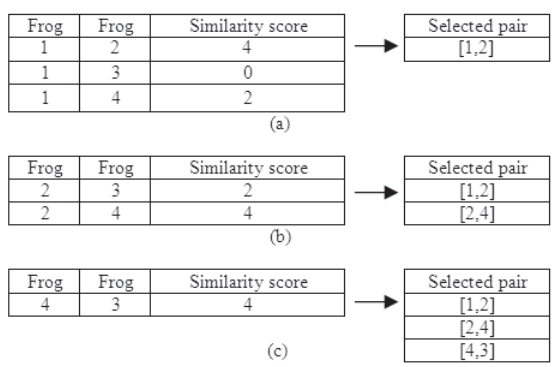

Fig. 3. Example of the process of assigning frogs to slots of a roulette wheel.

After all frogs are allocated slots in the roulette wheel, the overlapping area between the two adjacent frogs is determined. The size of the overlapping area is positively correlated with the similarity between the two adjacent

frogs. The more similar the two adjacent frogs is, the bigger the size of the overlapping area. Fig 4 shows the fuzzy roulette wheel of the example in Table 2.

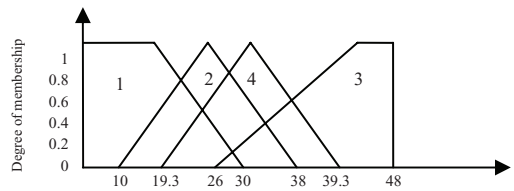

Fig. 4. Fuzzy roulette wheel

In the selection of frogs to form a sub-memeplex, first, a random number is generated. Then the membership degree to which a generated random number belongs to each frog is calculated. The one with the highest membership degree is chosen. This selection process continues until a total of q frogs has been selected.

3.3. Improve the performance of the worst frog of each memeplex

The proposed SLFA-FS model employs the crossover and the mutation operators to improve the quality of the solutions. Crossover operator plays an important part in improving the worst frog in each memeplex. After forming a sub-memeplex within each memeplex, the top two ranking frogs in a sub-memeplex are selected for crossover. Once the crossover points are defined, the Precedence Operation Crossover (POX) is adopted to develop a new offspring (Eq. (4)). If the new offspring is fitter than the worst frog in the corresponding memeplex, the worst frog will be replaced by this new offspring . If not, the crossover between the best frog in the population and the second rank frog in the sub-memeplex is performed (Eq. (5)).

$$Offspring = Fsbest \oplus F2s \tag{4}$$

$$Offspring = Fgbest \oplus F2s \tag{5}$$

where F2s is the position of the second rank frog in the sub-memeplex.
$\oplus$ is the crossover denotation.
Fsbest is the position of the best frog in the sub-memeplex.
Fgbest is the position of the best frog in the population.

Best frog :

Job	2	1	1	2	1	2
Machine	M1	M1	M2	M2	M1	M1

Second rank frog :

Job	2	2	1	1	1	2
Machine	M3	M1	M2	M2	M1	M1

Offspring :

Job	2	1	1	2	1	2
Machine	M3	M1	M2	M1	M1	M1

Fig. 5. Example of the crossover operation

Once again, if the new offspring is fitter than the worst frog in the corresponding memeplex, the worst frog will be replaced by this new offspring. Following the crossover operation, the local search, which will be described in section 3.5, is performed on the new frog. However, if the new offspring is still worse than the worst frog in the corresponding memeplex, the mutation operator is employed to perform the mutation on the second rank frog in the sub-memeplex. Mutation operator is used in order to maintain the diversity of the population and to overcome premature convergence. The mutation procedure is as follows: 1) Select the machine that processes the last operation of the schedule. 2) List all operations that are processed on this machine. 3) Randomly select one operation from the list. 4) Re-assign the new machine to the selected operation. 5) Determine the fitness value of the

mutated frog. If the mutated frog is fitter than the worst frog in the corresponding memeplex, the worst frog will be replaced by this mutated frog. Then the mutated frog is further improved by the local search. However, if the mutated frog is not fitter than the worst frog in the corresponding memeplex, a new frog is randomly generated to replace that worst frog. This process continues until all m memeplexes are processed.

3.4. Shuffle the memeplexes

All the memeplexes are combined together. Then the population is sorted in order of decreasing performance value. Next, the termination criterion is checked. If it is met, the algorithm is terminated. If not, the population is repartitioned into memeplexes and the loop continues.

3.5. Local search procedure

The local search procedure used in this paper further improves the solution by swapping the operations within a block on the critical path. The critical path is the longest route from source to sink in the disjunctive graph. It is possible to decompose the critical path into a number of blocks (a block is a maximal sequence of adjacent critical operations that is processed on the same machine). Fig 6(a) shows the Gantt chart of one of the resulted solutions $O_{1,1}$, $O_{2,1}$, $O_{1,2}$, $O_{2,2}$, $O_{2,3}$, $O_{1,3}$. In Fig 6(a), the critical path is $O_{2,1}$-$O_{2,2}$-$O_{2,3}$-$O_{1,3}$. There is only one block on this critical path, which is $O_{2,3}$-$O_{1,3}$. By swapping the sequence of operations $O_{2,3}$ and $O_{1,3}$, the makespan of the solution is decreased from 19 to 17 as shown in Fig 6(b).

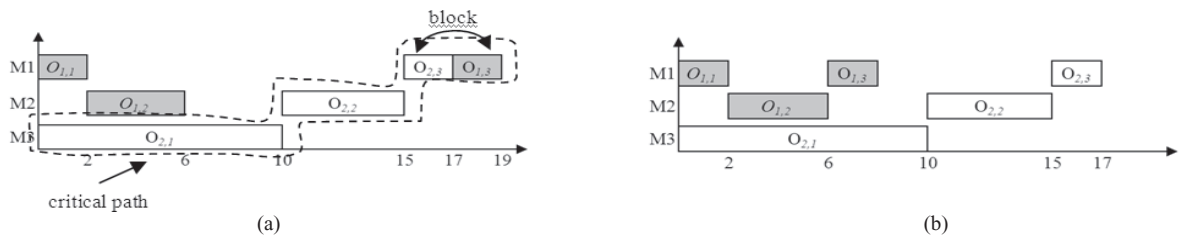

Fig. 6. Local search procedure

4. Experimental Results

The proposed algorithm was tested on 10 benchmark problems taken from [13]. The performance of the proposed algorithm is compared to that of the previous research work by Brandimarte [14]. The system parameters of SFLA-FS are set as follows: F = 50, 100; N = 100; m = 10% of F; n = F/m; G = 100; and q = 2/3 of n. The percentage relative error from the best known solution (RE) is used as a measure to evaluate the performance of the proposed method.

Table 3 shows the experimental results of our SFLA-FS in comparison to the tabu search of Brandimarte [14]. The results indicate that the SFLA-FS algorithm outperforms Brandimarte's approach in 6 out of 9 problems.

5. Conclusion

This paper proposes a modification to the shuffled frog leaping algorithm for solving the flexible job-shop scheduling problem (FJSP). The proposed model is a combination of the shuffled frog leaping algorithm and the fuzzy logic concept. The experimental results show that the performance of the shuffled frog leaping algorithm is significantly improved when a new fuzzy roulette wheel selection is used to construct a sub-memeplex, instead of the rank selection typically used in the original shuffled frog leaping algorithm. Our future work will deal with multi-objective FJSP, a much more complex version of the FJSP.

Table 3. Experimental results

Problems	R x S	LB	Brandimarte [14]			SFLA-FS		
			Popsize	Cmax	RE (%)	Popsize	Cmax	RE (%)
MK01	10 x 6	36	100	42	16.67	50	41*	13.89
MK02	10 x 6	24	100	32	33.33	100	30*	25.00
MK04	15 x 8	48	100	81	68.75	50	68*	41.67
MK05	15 x 4	168	100	186	10.71	100	181*	7.74
MK06	10 x 15	33	100	86	160.61	100	75*	127.27
MK07	20 x 5	133	100	157*	18.05	100	159	19.55
MK08	20 x 10	523	100	523*	0.00	100	523*	0.00
MK09	20 x 10	299	100	369*	23.41	50	369*	23.41
MK10	20 x 15	165	100	296	79.39	100	272*	64.85
Average Relative Error					45.66			35.93

References

1. D.E. Goldberg, Genetic Algorithms in Search Optimisation and Machine Lerning, Addison-Wesley, Reading, Massachusetts, 1989.
2. F. Glover, Tabu Search-Part I, Operations Research Society of America, J. Comput., (1989) 4 – 32.
3. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by Simulated Annealing, Sci., vol. 220 (1983) 671-680.
4. J. Kennedy and R. Eberhard, Particle Swarm Optimization, Phys.Rev.,B.(1995) 13:5344-5348.
5. L. Zadeh, Fuzzy sets, Information and Control, 8(3) (1965) 338-353.
6. X.-H. Luo, Y. Yang and X. Li, Solving TSP with shuffled frog-leaping algorithm, Proc. ISDA (2008) 228 - 232
7. Ziyang Zhen, Daobo Wang, Yuanyuan Liu, Improved Shuffled Frog Leaping Algorithm for Continuous Optimization Problem, IEEE Congress on Evolutionary Computation , CEC, Norway, Trondheim, 2009.
8. Z. Y. Zhen, Z. S. Wang, Z. Gu , Y. Y. Liu, A novel memetic algorithm for global optimization based on PSO and SFLA, Lecture Notes in Computer Science, Springer (2007) 127-136.
9. M.M. Eusuff, K.E. Lansey, Optimization of water distribution network design using the shuffled frog leaping algorithm, in J Water Resource Plan Manage (2003) 210–225.
10. M.M. Eusuff, K.E. Lansey, F. Pasha, Shuffled Frog-Leaping Algorithm: A Memetic Meta-heuristic for Discrete Optimization, Engineering and Technology, Mathematics and Optimization (2006) 129–154.
11. S.N. Sivanandam, S. Sumathi and S.N. Deepa, Introduction to Fuzzy Logic using Matlab, Springer-Verlag, Berlin Heidelberg, 2007.
12. Z. Chaoyong, R. Yunqing and Li. Peigen, An effective hybrid genetic algorithm for the job shop scheduling problem, Int J Adv Manuf Technol 39 (2008) 965–974.
13. M. Mastrolilli, Flexible Job Shop Problem, http://www.idsia.ch/~monaldo/fjsp.html.
14. P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, Annals of Operations Research 22 (1993) 158 – 183.